最近公共祖先Least Common Ancestor

2019年7月9日 22:55

-
- 最近公共祖先LCA

一. LCA
  1. CA公共祖先z：既是x祖先又是y祖先的结点z
  2. LCA最近公共祖先z：上述z中深度最大的z
  3. 向上标记法：从x向上找到根，标记走过的结点，从y也向上走，找到首个标记

二. 树上倍增法
  1. 数据结构：
     i. 令f[x][k]存x结点的第2^k次方上层的结点
     ii. 如f[x][0]存父节点
     iii. 先O（NlogN）的预处理，再O（logN）地求LCA
  2. 思路：
     i. 设y是更深的点，先将y拉到与x等深
     ii. 再一起约好向上找2^i层，如果不是CA，则一起向上2^i层
     iii. 易知这种转移方法就是二进制拆分法，到2^0时可以恰好拆到LCA下一层
     iv. 此时上一层的结点就是LCA了
  3. 预处理：

```
 i. queue<int>q;
    int dep[MN]; //层数，从1开始
    int f[MN][20];      //i向上2^j层的父节点
    int lg[MN],t;//lg[i]= log2(i); t=max{lg[i]}
ii. void bfs(){
        q.push(1);
        dep[1]=1;
        while(q.size()){
            int x=q.top();
            q.pop();
            for(int i=fst[x]; i; i=nxt[i]){
                int y= to[i];
                if(dep[y])    //无向树中忽视指向父亲的边
                    continue;
                dep[y]= dep[x]+1;
                f[y][0]= x;
                for__(j,1,t)
                    f[y][j]=f[ f[y][j-1] ] [j-1];
                q.push(y);}}}
iii. for__(i,1,n)
        lg[i]= logw(i);
    t= lg[n];
```

  4. 查询：

```
 i. int lca(int x,int y){
        if(dep[x]>dep[y])   //保证y是更深的结点
            swap(x,y);
        rof__(i,t,0) //从最高处往下找道与x等高为止
            if(dep[f[y][i]] >=dep[x])
                y=f[y][i];
        if(x==y)
```

```
        return x;
    rof__(i,t,0) //从最高处往下找第一个不相同的
        if(f[x][i]!=f[y][i])
            x=f[x][i],
            y=f[y][i];
    return f[x][0];}
```

-
- 例题

一. 双向地图中任两点间最大载重（=最小边权的最大值）（USSTD3I）

1. 引：为排除无关小权边，给所有边按权排降序，再kruskal求最大生成树
2. 转：在预处理祖先同时处理最小边权，找lca过程中遇到的最小边即为答案
3. void dfs(int x,int y){ //初始化各连通分支，进入dfs前已更新好mv[y][0]

```
    vst[y] = 1;
    dep[y] = dep[x] +1;
    f[y][0] = x;
    for__(i,1,lg[dep[y]])      //初始化向上的信息
        f[y][i]= f[ f[y][i-1] ][i-1],
        //上2^i父亲 = 上2^i-1父亲的上2^i-1父亲
        mv[y][i]= min(mv[y][i-1], mv[ f[y][i-1] ][i-1]);
        //上2^i最小权 =上 2^i-1最小权 或者 上2^i-1父亲更上2^i-1最小权
    for(int i= fst[y]; i; i= nxt[i])          //初始化向下的信息
        if(!vst[to[i]])
            mv[to[i]][0] = val[i],     //单边，最小权只能是该边的权
            dfs(y,to[i]);}
```

4. int solve(int x,int y){

```
    int ret=0x3f3f3f3f;
    if(dep[x]>dep[y])
        swap(x,y);      //接下来三行建立在y比x深的前提下
    while(dep[y]>dep[x])    //使y上移到与x同层
        ret= min(ret, mv[y][ lg[ dep[y]-dep[x] ] ]),
//cout<<"dy"<<dep[y]<<" dx"<<dep[x]<<" lg"<<lg[dep[y]-dep[x]]<<" ret"<<ret<<'\n',
        y= f[y][lg[ dep[y]-dep[x] ] ];
    if(x==y)
        return ret;
    rof__(i,lg[dep[x]],0)     //向上倍增求lca，顺便更新ret
        if(f[x][i]!=f[y][i]) //需要上移
            ret= min(ret, min(mv[x][i], mv[y][i])),
            x=f[x][i],
            y=f[y][i];
    return min(ret, min(mv[x][0], mv[y][0]));        //注意此时xy在lca正下
}
```

5. for__(x,1,n)

```
    if(!vst[x])        //新的连通分支的根
        mv[x][0]=0x3f3f3f3f,    //根没有父节点，距离无穷大
        dfs(x,x);//在dfs时更新dep[x]=1
//      for_(i,1,10) for_(j,0,4) cout<<"mv "<<i<<" "<<j<<" = "<<mv[i][j]<<"\n";
scanf("%d",&q);
while(2 == scanf("%d%d",&u,&v))
    if(fd(u)!=fd(v))
        puts("-1");
```

```
else
        printf("%d\n",solve(u,v));
```